IN THE SPECIFICATION

Please replace the abstract of the disclosure with the following replacement abstract of the disclosure:

--A distributed system enables the sharing of structured and unstructured knowledge using a publish and subscribe pattern. An evolving ontology of knowledge types is maintained within the system. The system associates received data with a datatype that has a format that is readable by other users of the system, then shares the datatype with relevant subscribers on the system. Upon receiving the datatype, the subscribers can also access the data, which is maintained separately from the datatype. The data is associated with meta data that is in a format that can be recognized by users of the system. Thus, the data itself does not have to be in a globally recognizable format. Over time, more information can be driven into the meta data, so that data processors know less and less about the original format of the data, and the data can also be converted to other formats. ~~Methods, systems, and articles of manufacture consistent with the present invention provide for evolutionary development of intellectual capital. A data instance is asynchronously received in a first format. A copy of the data instance is asynchronously received in a second format different than the first format. A datatype of a third format is provided for the data instance and the copy of the data instance. Each datatype has a metadata in the third format that describes the respective data instance and a reference in the third format to the respective data instance. The data instances are maintained separately from the datatypes. The third format is recognizable to a subscriber of the data instances to enable the subscriber to concurrently process the data instance in the first format and the copy of the data instance in the second format. The data instance in the first format is converted to the second format.~~--

Please replace the paragraph beginning at page 9, line 18, with the following replacement paragraph:

--Additional devices can also be connected to the network as part of the system. In the depicted example, a legacy storage system 130, which has a legacy data storage device 132, is connected to the network. The system can access intellectual capital and data stored on the legacy storage system. Intellectual capital data is also stored on a file server 150 connected to the network.

3

File server 150 includes a file manager program and a file storage. Each of these components of the system will be described in more detail below.--

Please replace the paragraph beginning at page 21, line 20, with the following replacement paragraph:

--If the registration manager determines that the user wants to delete a datatype (step ~~620~~ 622), then the registration manager deletes the datatype from the registry (step 622). To do so, the registration manager issues a request, such as an SQL request, to the properties RDBMS associated with the datatype to delete a table for the datatype in the properties database. Also, the registration manager issues a request, such as an S1MQ request, to the bus manager to delete the message topic associated with the datatype. And the registration manager issues a request to the file server manager to deregister the datatype. Alternatively, the registration manager can keep the datatype in the registry, but mark the datatype as invalid by setting the datatype status field to INVALID.--

Please replace the paragraph beginning at page 24, line 15, with the following replacement paragraph:

--After the registration manager receives the client information from the user, the registration manager generates the registration manager generated fields, as shown in Table 4, including a password for the client (step 710).--

Please replace the paragraph beginning at page 25, line 1, with the following replacement paragraph:

--The registration manager then checks whether the new or modified client is valid (step 716 ~~720~~). To do this, the registration manager determines whether the client information is complete and the client name is unique. The registration manager then commits the client to the registry (step 718).--

Please replace the paragraph beginning at page 27, line 12, with the following replacement paragraph:

--A functional block diagram of the client module and associated clients is shown in Figure

4

10. Although three types of clients are shown with a single client module, this is to illustrate that each of those client types can be associated with the client module. A different instance of the client module, however, is instantiated for each client. The client module has a client module Application Programming Interface (API) <u>1002</u>, which provides access to a developer to data and intellectual capital available on the system. The API is, for example, a Java® API.--

Please replace the paragraph beginning at page 37, line 3, with the following replacement paragraph:

--The storage controller can operate in three operating modes: local mode, remote mode, and legacy mode. Figure 18 depicts a functional block diagram of the storage controller operating in local mode. And Figure 19 depicts a functional block diagram of the storage controller operating in remote mode. Depending on whether the storage controller 225 is operating in local mode or remote mode, various functional components are illustrated. The storage controller interface 1802 exposes an storage controller API to the client module. The local mode plug-in 1804 interfaces with the JDBC interface 1806 and HTTP interface 1808 and manages the storage and delivery of data. The remote mode plug-in 1902 encodes and decodes the requests from the storage controller interface into document form for HTTP transmission and reception. The remote server 1906 is similar to the local mode plug-in in that it interfaces with the JDBC interface 1806 and HTTP interface 1808, and it encodes and decodes eXtensible Markup Language documents. The JDBC interface 1806 manages the interface with the properties database 250. The HTTP interfaces 1808, 1904 and 1910 interface between the storage controller 225 and the file server <u>150</u> ~~152~~, and between the storage controller 225 and the remote server 1906. Each of these functional components will be described in more detail below.--

Please replace the paragraph beginning at page 44, line 33, with the following replacement paragraph:

--Figure 24 depicts a functional block diagram illustrating how a datatype property mapping is achieved with the datatype mapping editor. Initially, a user enters ~~a draws~~ a map of the required properties for the datatype. The sources 2402 of the datatype, such as the document metadata and

5

SQL table fields, are then isolated. The user then builds a query that will allow the sources to be queried based on the values coming in from the legacy storage controller.--

Please replace the partial paragraph beginning at page 45, line 16, with the following replacement paragraph:

--The construction of the datatype body is managed in two ways. First ~~Firs~~, the queries are designed to extract the data components of the body. The results of these queries are then organized within the body as components, as shown in the following illustrative example:--

Please replace the paragraph beginning at page 45, line 30, with the following replacement paragraph:

--In addition to bringing in legacy data into the system through the legacy storage controller, the system can also acquire other external data into the system through the external data input manager. The external data input manager is an input gateway for external data to the system. It ~~Its~~ wraps and formats an incoming datatype in such a way that the data can be published and used in the system. Each datatype that is external has its own external data input manager. The system is defined in this manner because of the individual data instance specific variables and the tight coupling the external data input manager will have with the specific data type. A functional block diagram of external data input managers 2502 and 2504 receiving external data instances 2506 and 2508 and publishing to the messaging bus 2510 is shown in Figure 25. As shown, the external data input managers 2502 and 2504 communicate with the bus via client managers 2512 and 2514.--

Please replace the paragraph beginning at page 46, line 11, with the following replacement paragraph:

--Figure 26 depicts a flow diagram of the illustrative steps performed by the external data input manager. One having skill in the art will appreciate that this is one illustrative implementation of the external data input manager, and that its implementation will be influenced by the type and frequency of the data input being managed. First, the external data input manager receives an external data instance from a data source (step 2602 ~~2606~~). This can be done, for example, by receiving an electronic mail in an electronic mail queue that is periodically checked by the external

data input manager.--

Please replace the paragraph beginning at page 47, line 19, with the following replacement paragraph:

--This provides for complex chaining of passive intellectual capital that is influenced by active intellectual capital. Accordingly, problems with customer systems can be mapped to the intellectual <u>capital</u> quickly and dynamically. Further, new clients can be added to the system without the need for versioning the whole system. Therefore, dynamic solution paths through the system can be reused.--